Assignment - 2
(Design and Code Size)

Course Code	:	SE 3204
Course Title	:	Software Metrics

Submitted To:

Dipok Chandra Das
Assistant Professor,
Institue of Information Technology,
Noakhali Science and Technology University (NSTU)

Team Members:

1.	Md. Armanur Rashid	ASH1925013M
2.	Sourav Debnath	ASH1925022M
3.	Sourav Barman	ASH1925030M

Noakhali Science and Technology University (NSTU)

Contents

Figures

Code Size

Program code is an integral component of software. Such code includes source code, intermediate code, byte code, and even executable code. We look at approaches for directly measuring code size. We must take great care to clarify what we are counting and how we are counting it.

Number of LOC

Definition: Counting number of physical lines (including blank lines, comment lines).

Type: Automated

Value: 2376

Number of CLOC

Definition: Counting the commented lines of code. Comment can be single line comment (//) and multi-line comment (/**/).

Type: Automated

Value: 123

Number of NCLOC

Definition: Counting all lines excluding blank lines and commented lines of code. Called effective lines of code.

Type: Automated

Value: 2253

Density of Comment

Definition: Can be derived by: CLOC / (NCLOC + CLOC)

Type: Automated

Value: 5.2%

Average LOC

Definition: Can be derived by: LOC / Total Class

Type: Automated

Value: 113

Number of word

Definition: Number of characters in the program text.

Type: Automated

Value: 7446

Number of bytes of computer storage

Definition: Number of bytes used in the computer storage for the program text.

Type: Automated

Value: 541 KB

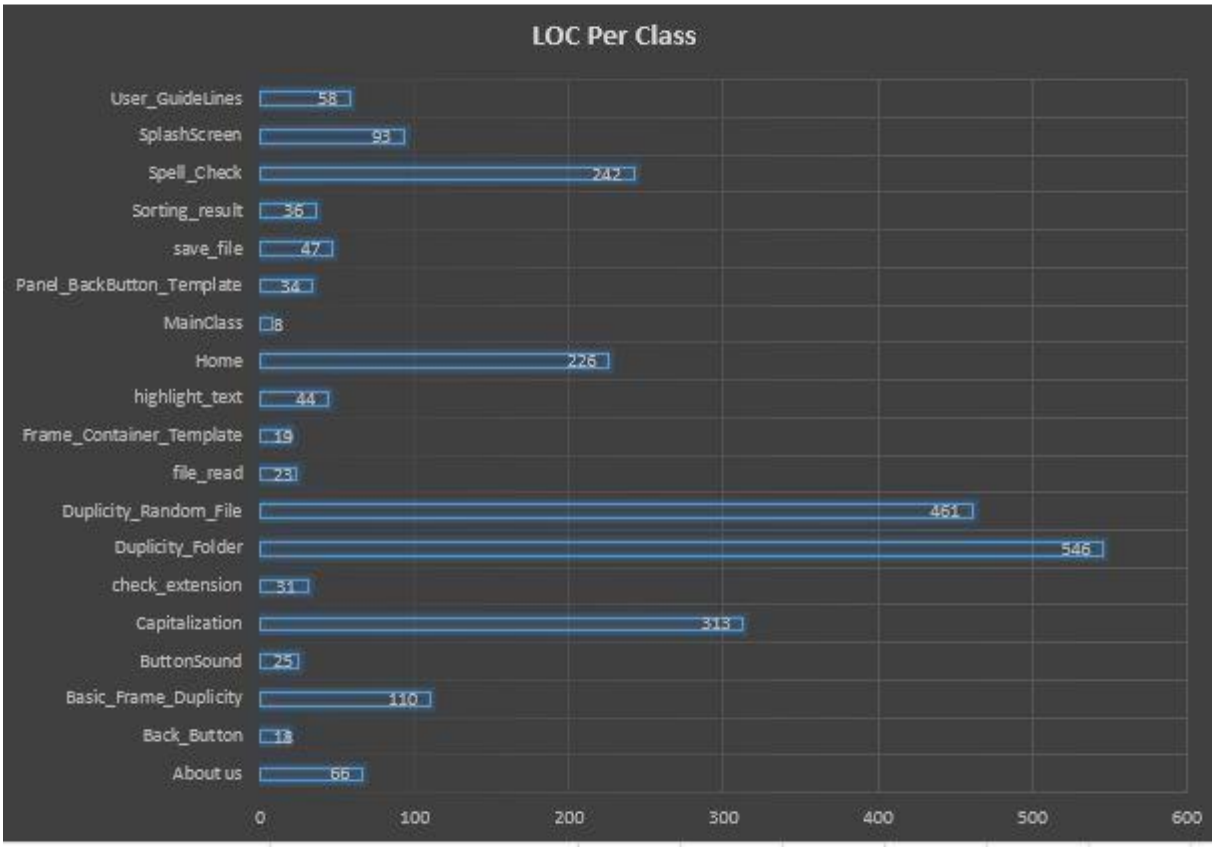

Figure 1: LOC Per Class

Halstead's Approach

Halstead's software science attempted to capture attributes of a program that paralleled physical and psychological measurements in other disciplines. He began by defining a program P as a collection of tokens, classified as either operators or operands.

$\mu 1$ = Number of unique operators

$\mu 2$ = Number of unique operands

N1 = Total occurrences of operators

N2 = Total occurrences of operands

The basic metrics for these tokens are the following:

Measurement	Statistics
Unique Operator, $\mu 1$	12
Unique Operand, $\mu 2$	89
Total Operator, N1	172
Total Operand, N2	379

The length of P is defined to be N = N1 + N2, while the vocabulary of P is $\mu 1 = \mu 2 + \mu 3$. The volume of a program, akin to the number of mental comparisons needed to write a program of length N or the minimum number of bits to represent a program, is

$$\begin{aligned} N &= N1 + N2 \\ &= 172 + 379 \\ &= 551 \end{aligned}$$

While the vocabulary of P is:

$$\begin{aligned} \mu &= \mu 1 + \mu 2 \\ &= 12 + 89 \\ &= 101 \end{aligned}$$

The volume of a program, akin to the number of mental comparisons needed to write a program of length N or the minimum number of bits to represent a program, that is

$$\begin{aligned} V &= N \times \log_2 \mu \\ &= 551 \times 6.66 \\ &= 3669.66 \end{aligned}$$

Design Size

We can measure the size of a design in a manner similar to that used to measure code size. We will count design elements rather than LOCs. The elements that we count depend on the abstractions used to express the design, and the design aspects of interest. Thus, the appropriate size measure depends on the design methodology, the artifacts developed, and the level of abstraction. Thus, we will measure size in terms of packages, design patterns, classes, interfaces, abstract classes, operations, and methods.

Number of Sub Packages

Definition: Counting the number of sub packages under a package.

Type: Manual

Value: 1

Number of Class

Definition: Counting the number of classes used.

Type: Manual

Value: 21

Number of Interface

Definition: Counting the number of interfaces used.

Type: Manual

Value: 0

Number of Methods

Definition: Counting the number of method used.

Type: Manual

Value: 63

Number of Average Method Per Class

Definition: Counting the number of method per class.

Type: Manual

Value: 3

Number of Design Principle

Definition: Counting the number of design principle used.

Type: Manual

Value: 2

Number of Design Pattern

Definition: Counting the number of design principle used.

Type: Manual

Value: 1

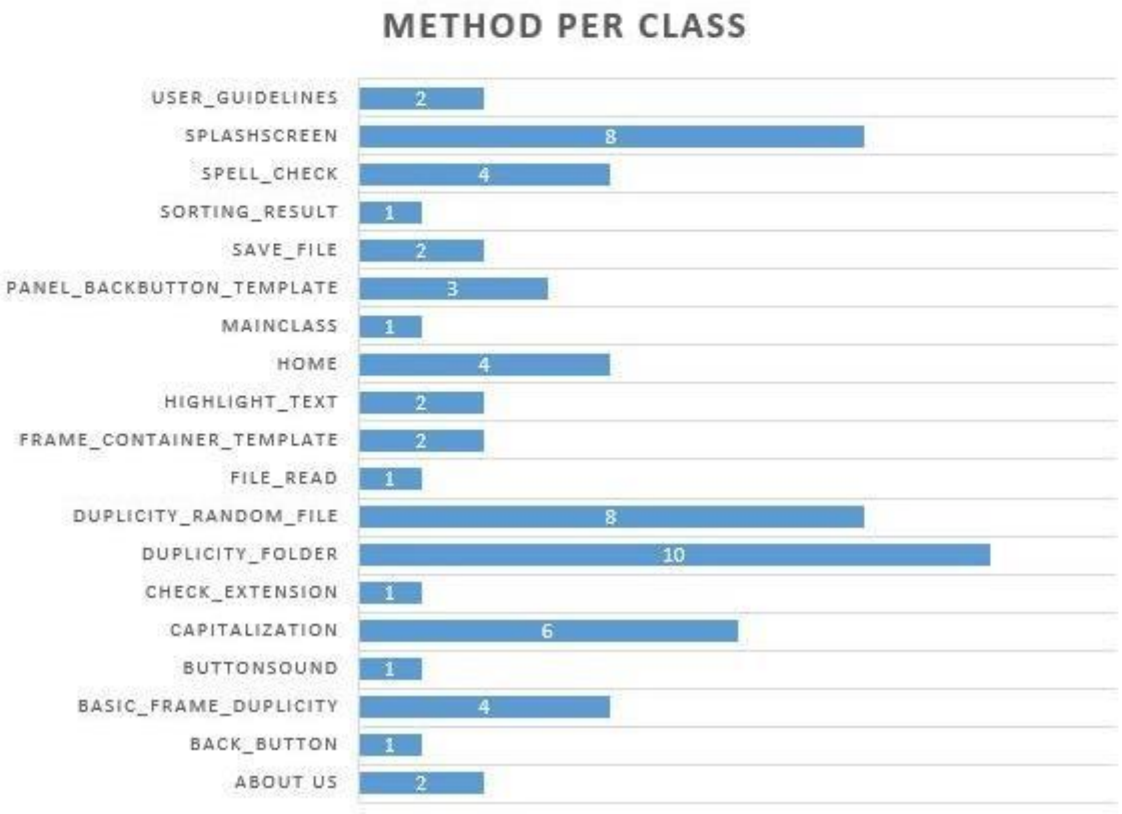

Figure 2: Method Per Class

Halstead’s Java Code

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;
public class HalsteadApproach {
    public static String REGEX = "^[A-za-z_$]{1,}[A-za-z0-9_$]{0,}(.java)$";
    public static String NAMING = "[=(.\s]*[A-za-z_$]{1,}[A-za-z0-9_$]{0,}[=\s;)]*$";
    static String []opers={"=", "+=" ,"-=", "*=", "/=" ,"%=", "&=" ,"^=" ,"|=" ,"<<=" ,">>=",
">>>=" ,"?" ,":" ,"||" ,"&&", "|" ,"^" ,"&" ,"==" ,"!=", "<" ,">" ,"<=" ,">=" ,"<<" ,">>"
,">>>" ,"+" ,"-"          ,"*" ,"/" ,"%", "~", "!"};

    static List<String> operators= Arrays.asList(opers);
    static int operatorsCount=0;
    static int  operandCount=0;
    static List<String> operatorList;
    public static int[] listFilesForFolder(final File folder) throws FileNotFoundException {
        operatorList = new ArrayList<>();
        List<String> operandList= new ArrayList<>();
        int []counts= new int[4];
        for (final File fileEntry : folder.listFiles()) {
            if (fileEntry.isDirectory()) {
                listFilesForFolder(fileEntry);
            } else {
                if(fileEntry.getName().matches(REGEX)){
                    File f= new File(fileEntry.getPath());
                    Scanner collect=new Scanner(f);
                    while(collect.hasNext()){
                        String str=collect.next();
                        for(String oprs: operators){
                            if(str.contains(oprs)){
                                operatorList.add(oprs);
                            operatorsCount+=1;
                            }
                    }
                    }
                }
                }
        }
        counts[0]=new HashSet<String>(operatorList).size();
        counts[1]=0;
        counts[2]=operatorsCount;
        counts[3]=operandCount;
        return counts;
    }
}

class Main{
    public static void main(String[] args) throws FileNotFoundException {
        final File folder = new File("C:\\Users\\Desktop\\SPL1");
        int arr[]=HalsteadApproach.listFilesForFolder(folder);
        System.out.println("Unique Operators: "+arr[0]);
        System.out.println("Total Operators: "+arr[2]);
    }
}
```